In the Claims:

Please amend the claims as follows:

1. (Canceled)

2. (Canceled)

3. (Canceled)

4. (Canceled)

5. (Canceled)

6. (Canceled)

7. (Canceled)

8. (Canceled)

9. (Canceled)

10. (Canceled)

11. (Currently Amended) A method for efficiently handling high contention locking in a multiprocessor computer system, comprising:

organizing at least some of the processors into a hierarchy;

providing a lock selected from the group consisting of: an interruptible lock, and a lock which waits using only local memory;

processing the lock responsive to the hierarchy; and

~~The method of claim 1, further comprising~~ maintaining a release flag for a group of processors to prevent races between acquisition and release of the lock.

12. (Currently Amended) A method for efficiently handling high contention locking in a multiprocessor computer system, comprising:

organizing at least some of the processors into a hierarchy;

providing a lock selected from the group consisting of: an interruptible lock, and a lock which waits using only local memory;

processing the lock responsive to the hierarchy; and ~~The method of claim 1, further comprising~~ maintaining a handoff flag for a group of processors to grant the lock to a processor requesting an unconditional lock from a processor requesting a conditional lock.

13. (Canceled)

14. (Canceled)

15. (Canceled)

16. (Canceled)

17. (Canceled)

18. (Canceled)

19. (Canceled)

20. (Canceled)

21. (Canceled)

22. (Canceled)

23. (Canceled)

24. (Canceled)

25. (Canceled)

26. (Canceled)

27. (Canceled)

28. (Canceled)

29. (Canceled)

30. (Canceled)

31. (Canceled)

32. (New) The method of claim 11, wherein the processing step conditionally acquires the lock.

33. (New) The method of claim 11, wherein the processing step returns a failure to grant the lock if the lock is not immediately available.

34. (New) The method of claim 11, wherein the processing step unconditionally acquires the lock.

35. (New) The method of claim 34, wherein the processing step spins on the lock until the lock is available.

36. (New) The method of claim 34, further comprising allowing system interrupts while spinning on the lock

37. (New) The method of claim 11, wherein the processing step unconditionally releases the lock.

38. (New) The method of claim 11, wherein the processing step the processors spin on private memory.

39. (New) The method of claim 11, wherein the hierarchy includes a data structure having a bit mask indicating which processors of a group are waiting for the lock.

40. (New) The method of claim 11, wherein the hierarchy includes a data structure having a bit mask indicating which groups of processors have processors waiting for the lock.

41. (New) the method of claim 11, further comprising maintaining a handoff flag for a group of processors to grant the lock to a processor requesting an unconditional lock from a processor requesting a conditional lock.

42. (New) The method of claim 12, wherein the processing step conditionally acquires the lock.

43. (New) The method of claim 12, wherein the processing step returns a failure to grant the lock if the lock is not immediately available.

44. (New) The method of claim 12, wherein the processing step unconditionally acquires the lock.

45. (New) The method of claim 44, wherein the processing step spins on the lock until the lock is available.

46. (New) The method of claim 44, further comprising allowing system interrupts while spinning on the lock.

47. (New) The method of claim 12, wherein the processing step unconditionally releases the lock.

48. (New) The method of claim 12, wherein the processing step the processors spin on private memory.

49. (New) The method of claim 12, wherein the hierarchy includes a data structure having a bit mask indicating which processors of a group are waiting for the lock.

50. (New) The method of claim 12, wherein the hierarchy includes a data structure having a bit mask indicating which groups of processors have processors waiting for the lock.

51. (New) The method of claim 12, further comprising maintaining a release flag for a group of processors to prevent races between acquisition and release of the lock.